

Hardened Anti-Reverse Engineering System

@JacobTorrey

Assured Information Security (@ainfosec)

March 2015

Outline

Introduction

Background

How HARES Works

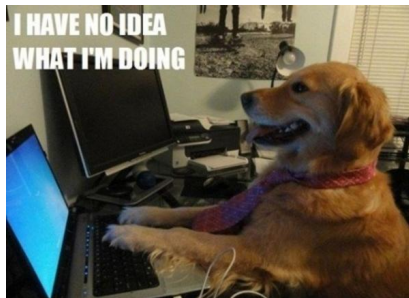
Results

Analysis & Future Work

Conclusions

Who am I?

- Advising Research Engineer at Assured Information Security (words are my own)
- Site lead for Denver, CO office
- Leads low-level Computer Architectures research group
- Plays in Intel privilege rings ≤ 0
- Ultra-runner, ultra-cyclist & mountaineer



Overview

- HARES provides the ability to execute *fully-encrypted* binaries
- Minimal performance impact at ~2% on unmodified Intel Core-i series processor
- Prevents key or instruction leakage even to compromised OS kernel
- Can protect *unmodified* binary applications *without* source or recompilation

Problem Statement

- Application code can be used to develop attacks
- Algorithms exposed to copying or theft
- Code can be reused for unintended purposes (ROP)



Current State-of-the-Art

- Obfuscation and packers — Analysis tools and live debugging can recover instructions
- VM-based obfuscation — Can still be mapped to x86 and impacts performance
- Encrypting entire OS with trusted boot — Only prevents against offline attacks

AES-NI

- In response to software-based caching attacks on AES, Intel released instruction set to support AES
- Hardware logic is faster, and more protected
- Supports 128-bit and 256-bit AES
- Provides primitives, still requires engineering to make a safe system on top of these

TRESOR

- Uses AES-NI and CPU debug registers to provide accelerated, cold-RAM resistant AES on Linux
- Key loaded early boot
- Kernel patch to prevent applications reading debug registers

TLB-Splitting

- Translation lookaside buffer (TLB) is a cache for virtual — physical address translations
- Used to improve paging performance
- Logically treated as single entity, *physically multiple components*
- Switches x86 platform from apparent Von Neumann to Harvard architecture:
 - Used by PaX/GRSecurity to emulate no-execute bit
 - Shadow Walker used for memory-hiding root-kit functionality

TLB-Splitting (cont.)

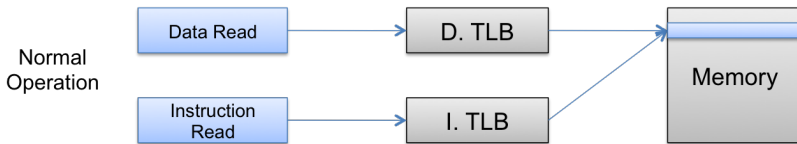


Figure 1: TLB during normal operation

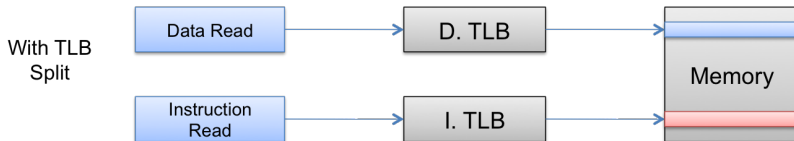


Figure 2: Split TLB

S-TLB

With Nehalem micro-architecture, an L2 cache was introduced, the S-TLB; breaks split-TLB assumptions

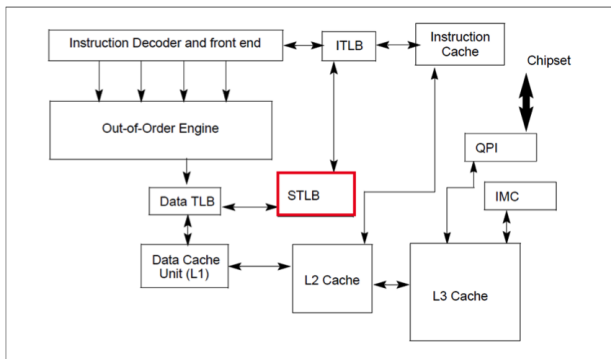


Figure 3: S-TLB

MoRE: Measurement of Running Executables

- DARPA Cyber Fast Track program
- Explored using TLB-splitting for measurement/integrity verification of interleaved application
 - Immutable code page (can repeatably measure in real-time)
 - Mutable data page (for variable isolation)
- Used EPT granular permissions to simulate a split-TLB on newer CPUs with S-TLB
- Thin-VMM to simulate Harvard architecture on per-process basis

VMX Thin-Hypervisor

- Loaded as Windows 7 kernel driver
- Based on vmcpu root-kit example
- No emulation of devices, OS retains direct access to HW
- Minimal performance impact
- Can use VMCS exit conditions to track certain architectural event

On-CPU AES

- “Hoists” TRESOR on-CPU AES into VMM
- Adds VMCS exit condition for debug register accesses (return NULL or silently discard write)
- Decrypts executable sections of program into *execute-only* memory



Process Creation Monitoring



- Registers call-back for process creation
- Notified before execution and during termination
- Parses PE and identifies regions to decrypt and perform TLB-split on
- Uses hyper-calls to begin or terminate VMM support

TLB-Splitting

- All data-fetch requests, even from application itself, directed via EPT/TLB to encrypted page
- All instruction fetches are directed to execute-only, decrypted, memory
- Must track Windows application memory management events (COW) and ensure EPT structures correspond with OS-level structures



Test Cases

- Calculation of π — Purposely inefficient power-series algorithm to approximate π
- Random Sort — Randomized CPU and memory access patterns to test the performance for non-consecutive cache-line accesses
- Coin-flip — Called many shared library functions to ensure compatibility
- Timers — Monitors performance impact as ratio of cycles to 'wall' time



Test Results

Overall, average performance impact was ~2%

Test Name	Average Execution Time (s)	Average HARES Execution Time (s)	Performance Impact (%)
Pi	28.173	28.600	1.515
Randomized Sort ¹	0.016	0.031	95.83
Coin Flip	1.778	1.809	1.762
Timers	15.013	15.023	0.069

Table 1: Performance Results for CLI Test-Suite

¹The randomized sort runs for such a short period that the initialization routine of the Windows process creation monitor almost doubles the execution time. If the TLB-splitting and periodic measurement is disabled, the run-time is still almost double. For this test, it is reasonable to assume that there is a constant increase of ~.015s for each test case run under the VMX hypervisor, and due to this test case's short duration it skewed the percentage.

Tested Applications

- Aforementioned synthetic test-suite
- Microsoft Windows Notepad
- Microsoft Windows Paint
- Microsoft Windows Calculator

Usability of HARES-protected applications was not noticeably impacted to end-user

Demonstration System

Specifications for the demonstration system to overcome some prototype limitations:

- Intel Core-i series processor with AES-NI; Windows 7 32-bit OS
- Single processor [numproc=1] (Our thin-VMM only supports single core currently)
- 2GiB RAM [truncatememory=0x80000000] (Windows kernel memory layout changes > 2 GiB, and too lazy to update hard-coded values)
- No PAE/DEP [nx=AlwaysOff and pae=ForceDisable] (Again, hard-coded memory layout code)

Demonstration

[ALT-TAB]

Engineering Challenges

- Mixed code & data in PE section
- Paging out of application memory
- COW/relocation of application



Overcoming Challenges: Mixed Code & Data

- Easily identifiable
 - Import tables, debug tables, etc., are easily parsed and excluded from encryption
- Not so easily identifiable
 - Single purpose strings and other small data are often stored adjacent to the code that uses them and are difficult to identify in compiled code
- Not a problem if source is available
 - Compiler options can be used to create read-only sections
- Binary only
 - Reverse engineering — time consuming & unreliable
 - Provisioning/Learning Mode — it works for proof of concept, but unreliable for large programs
 - Debug symbols

Overcoming Challenges (cont.)



- Uses `MmProbeAndLockPages()` to prevent OS page-out — limited non-paged pool
- VM Exit on CR3 change to re-walk page-tables to detect COW — update TLB-split pages via hyper-call

Security Benefits

Protects against:

- Reverse-engineering and algorithmic IP theft
- “Weaponization” of crash case into RCE
- Mining application source for ROP gadgets
- Harvard architecture resistant to code injection attacks

Weaknesses

- JTAG/ICE/XDM
- Memory Dumping
- DMA
- SMM/AMT
- Side-channels
- Emulation/VMM



Overcoming Weaknesses & Future Work

- VT-d/IOMMU
- Cache-as-RAM
- DRTM launch (e.g., Intel TXT)
- Combining with unique compilation



Unintended Use-Cases

**the grugq**

@thegrugq



Following

@semibogan @JacobTorrey end of (easy)
malware analysis :D



RETWEETS

10

FAVORITES

9



6:31 AM - 8 Jan 2015

- Offensive key management is a less-studied practice and more challenging/likely to get wrong
- Easier to use for defense than offense

AV Heuristics



SHA256: 1acbe6931408a46efc8f2dafed28c39a081641bde87ca15634a870b96e660a4b

File name: notepad.exe

Detection ratio: 4 / 57

Analysis date: 2015-03-25 23:25:53 UTC (1 minute ago)



Analysis

File detail

Additional information

Comments

Votes

Antivirus	Result	Update
AVG	Win32/Heur	20150326
Bkav	HW32.Packed.35C0	20150325
Rising	PE:Malware.XPACK-HIE/Heur1.9C48	20150325
Tencent	Trojan.Win32.YY.Gen.3	20150326
ALYac	✓	20150326

- However, notepad.exe (unencrypted) with a 1-bit change to remove from MS white-list *also* hits 4/57

Concluding Remarks

- Demonstrates viability of encrypted execution on existing, common hardware
- Significantly increases reversing difficulty with minimal performance impact
- Provides vulnerable legacy systems “breathing room” until appropriate fixes can be implemented
- Intel SGX will be an exciting hardware extension to the platform and should be explored

Acknowledgments

- Mark Bridgman (@c0ercion) for his work on this effort
- Mudge (@dotMudge) & DARPA for supporting the precursor work (MoRE)
- Loc Nguyen (@nocsi_) & Ryan Stortz (@withzombies) for their input from a reverse engineering perspective



References

Formal references can be found in the whitepaper for:

- GRSecurity PAGEEXEC
- Shadow Walker
- Intel SDM
- TRESOR & TRESOR-Hunt
- ARIUM website
- Self-hashing applications
- CoreBoot Cache-as-RAM

and more.

Thanks!

See you all next year at Syscan
2016!



Questions & Discussion

- Thanks for your attention!
- Any questions? *or* let the heckling begin!
- Whitepaper can soon be found on my Twitter profile (@JacobTorrey: pinned-tweet)